UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/813,599	03/31/2004	Gansha Wu	42339-198432	4361

26694 7590 11/28/2007

VENABLE LLP
P.O. BOX 34385
WASHINGTON, DC 20043-9998

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
11/28/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/813,599	WU ET AL.	
	Examiner	Art Unit	
	Aimee J. Li	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *31 August 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *31 March 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-20 have been considered. Claim 24 has been added withdrawn.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as filed 31 August 2007.

Specification

3. The amendment filed 09 April 2007 is objected to under 35 U.S.C. 132(a) because it introduces new matter into the disclosure. 35 U.S.C. 132(a) states that no amendment shall introduce new matter into the disclosure of the invention. The added material which is not supported by the original disclosure is as follows: the language has been amended from reciting "shared execution code" to "cascading execution code". The terms "shared" and "cascading" have significantly different meanings in the art. The Examiner is unsure where support for changing the terminology from "shared" to "cascading" is supported in the original specification and there were no clear arguments that stated and supported that this change in terminology is not new matter. Applicant is required to cancel the new matter in the reply to this Office Action.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 1-20 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not

described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. Please see the new matter objection to the specification above for further explanation.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindwer, U.S. Patent Number 6,298,434 (herein referred to as Lindwer) in view of Raz et al., U.S. Patent Number 6,606,743 (herein referred to as Raz) and in further view of Chan et al., U.S. Patent Number 5,734,908 (herein referred to as Chan).

8. Referring to claim 1, Lindwer has taught a method to execute an instruction on an operand stack, the method comprising:

a. Performing a stack-state-aware translation of the instruction to code to determine an operand stack state for the instruction (Lindwer column 11, lines 3-22) (the preprocessor moves items from registers to memory and adjusts the SP for the instructions);

b. Dispatching the instruction according to the operand stack state for the instruction (inherent); and

c. Executing the instruction (inherent).

9. Lindwer has not taught his translation including determining an entry point into execution code based on the stack state. Raz has taught a language accelerator that uses memory-mapped registers as a stack (Raz column 4, lines 7-12) among other aspects (Raz column 1, line 60 to column 2, line 10). Raz's accelerator translates foreign code to native code and uses memory instructions to implement the stack operations on the memory-mapped stack (Raz column 6, lines 33-38). Thus, the stack state will determine what instructions are used to implement stack operations (e.g. an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz column 6, lines 24-51, emphasis on lines 45-51) and the code is threaded (Raz column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention. Raz states that his method increases the speed at which Java code is executed (Raz column 2, lines 11-12). In addition, Raz's method can be readily implemented in any processor (Raz column 13-1 9), while Lindwer's cannot. Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's to be able to execute Java code faster and to be able to implement the method in any processor.

10. In addition, Lindwer has not taught cascading execution code, wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier, each tier comprising at least one computer-executable instruction, and wherein the execution comprises entering the cascading code at an entry tier indicated by the determined entry point and executing the entry tier and at least one tier below the entry tier. Chan has taught

a. Cascading execution code (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are cascaded because once the entry basic block is executed at least one of the following, lower basic blocks are executed.),

b. Wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic

block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are each tier.),

c. Each tier comprising at least one computer-executable instruction (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B), and

d. Wherein the execution comprises entering the cascading code at an entry tier indicated by the determined entry point and executing the entry tier and at least one tier below the entry tier (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks...";

column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are cascaded because once the entry basic block is executed at least one of the following, lower basic blocks are executed.).

11. A person of ordinary skill in the art at the time the invention was made would have recognized, and as taught by Chan, that the basic blocks more fully utilize processor resources (Chan column 1, lines 24-33 "...a software compiler that synthesizes code that more fully utilizes the resources..."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the basic block tiers of Chan in the device of Lindwer to improve resource utilization.

12. Referring to claim 2, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 1, said performing comprising:

a. Determining a number of operands on the operand stack before the instruction is executed (Lindwer column 11, lines 3-22) (It is inherent that this step will be taken in moving items from registers to memory and adjusting the SP for the instructions);

b. Determining a number of operands on the operand stack after the instruction is executed based on a number of operands that the instruction consumes and a

number of operands that the instruction produces (Linder column 11, lines 3-22); and

c. Inferring a number of shift operations required after execution of the instruction to maintain top-of-stack elements (Linder column 11, lines 3-22)

13. Referring to claim 3, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 2, wherein the number of shift operations required after execution of the instruction is based on the number of operands on the operand stack before the instruction is executed and the number of operands on the operand stack after the instruction is executed (Lindwer column 11, lines 15-22).

14. Referring to claim 4, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 2, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table (Lindwer column 6, lines 39-47) (The translation is based on a static table. Through the table, it is known how many operands will be used and how many will be placed back on the stack, and based on that is how many items are transferred to memory.).

15. Referring to claim 5, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 1, wherein the operand stack is a mixed-register stack (Lindwer column 11, lines 15-22).

16. Referring to claim 6, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 1, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of the operand stack after the execution of the instruction (Lindwer column 11, lines 15-22).

17. Referring to claim 7, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 6, wherein the top-of-stack elements comprise a register stack (Lindwer column 11, lines 15-22).

18. Referring to claim 8, Lindwer in view of Raz and in further view of Chan has taught the method according to claim 1, further comprising refilling the operand stack (Linder column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

19. Referring to claim 9, Lindwer has taught a system comprising:

a. An operand stack to execute an instruction (Linder column 11, lines 3-5); and

b. An interpreter to determine a state of the operand stack, translate the instruction into threaded code, and dispatch the instruction based on the state of the operand stack (Linder column 11, lines 3-22) (the preprocessor is the interpreter).

20. Lindwer does not teach his interpreter determining an entry point into execution code based on the stack state. Raz teaches a language accelerator that uses memory-mapped registers as a stack (Raz column 4, lines 7-12) among other aspects (Raz column 1, line 60 to column 2, line 10). Raz's accelerator translates Java code, in some embodiments, to native code and uses memory instructions implement the stack operations on the memory-mapped stack (Raz column 6, lines 33-38). Thus, the stack state will determine what instructions are used to implement stack operations (e.g. an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz column 6, lines 24-51, emphasis on lines 45-51) and the code is threaded (Raz column 4, lines 29-31). The implementation and advantages of multithreading is

well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention. Raz states that his method increases the speed at which Java code is executed (Raz column 2, lines 11-12). In addition, Raz's method can be readily implemented in any processor (Raz column 13-1 9), while Lindwer's cannot. Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's to be able to execute Java code faster and to be able to implement the method in any processor.

21. In addition, Lindwer has not taught cascading execution code. Chan has taught cascading execution code (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are cascaded because once the entry basic block is executed at least one of the following, lower basic blocks are executed.). A person of ordinary skill in the art at the time the invention was made would have recognized, and as taught by Chan, that the basic blocks more fully utilize processor resources (Chan column 1, lines 24-33 "...a software compiler that synthesizes code that more fully utilizes the resources..."). Therefore, it would have been obvious to a person of ordinary

skill in the art at the time the invention was made to incorporate the basic block tiers of Chan in the device of Lindwer to improve resource utilization.

22. Referring to claim 10, Lindwer in view of Raz and in further view of Chan has taught the system according to claim 9, wherein the operand stack is a mixed stack comprising a register stack and a memory stack (Linder column 11, lines 15-22).

23. Referring to claim 11, Lindwer in view of Raz and in further view of Chan has taught the system according to claim 10, wherein the register stack comprises at least one register to hold at least one respective top element of the stack and the memory stack comprises a contiguous memory region to hold the remaining elements of the operand stack (Linder column 3, lines 15-22).

24. Referring to claim 12, Lindwer has taught a machine accessible medium containing program instructions that, when executed by a processor, cause the processor to perform a series of operations comprising:

a. Translating a virtual machine instruction into threaded code based on an operand stack state of the virtual machine instruction (Linder column 11, lines 3-22);

b. Dispatching the virtual machine instruction according to the operand stack state (inherent); and

c. Executing the instruction (inherent).

25. Lindwer does not teach his translation including determining an entry point into execution code based on the stack state. Raz teaches a language accelerator that uses memory-mapped registers as a stack (Raz column 4, lines 7-12) among other aspects (Raz column 1, line 60 to column 2, line 10). Raz's accelerator translates Java code, in some embodiments, to native

code and uses memory instructions implement the stack operations on the memory-mapped stack (Raz column 6, lines 33-38). Thus, the stack state will determine what instructions are used to implement stack operations (e.g., an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz column 6, lines 24-51, emphasis on lines 45-51) and the code is threaded (Raz column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention. Raz states that his method increases the speed at which Java code is executed (Raz column 2, lines 11-12). In addition, Raz's method can be readily implemented in any processor (Raz column 13-1 9), while Lindwer's cannot. Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's to be able to execute Java code faster and to be able to implement the method in any processor.

26. In addition, Lindwer has not taught cascading execution code, wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier, and each tier comprising at least one computer-executable instruction. Chan has taught

a. Cascading execution code (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an

thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are cascaded because once the entry basic block is executed at least one of the following, lower basic blocks are executed.),

b. Wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier (Chan Abstract "...The system operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B – In regards to Chan, the basic blocks are each tier.), and

c. Each tier comprising at least one computer-executable instruction (Chan Abstract "...The system operates by selecting from the source code representation a basic

block pair comprising a source basic block and one or more target basic block..."; column 1, lines 36-48 "...The invention operates by selecting from the source code representation a basic block pair comprising a source basic block and one or more target basic blocks..."; column 4, line 64 to column 5, line 25 "...Multiple basic blocks are circular an thus eligible for instruction movement if one of the basic blocks (which becomes the source basic block) always executes after the other basic block or another basic block via a control loop (these become target basic blocks)..."; column 12, lines 31-59 "...a source BB **904**, and a basic block **906**..."; Figure 3; Figure 9A-9C; and Figure 10A-10B).

27. A person of ordinary skill in the art at the time the invention was made would have recognized, and as taught by Chan, that the basic blocks more fully utilize processor resources (Chan column 1, lines 24-33 "...a software compiler that synthesizes code that more fully utilizes the resources..."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the basic block tiers of Chan in the device of Lindwer to improve resource utilization.

28. Referring to claim 13, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 12, wherein the threaded code is based on an entry point into cascading execution code (Linder column 11, lines 3-22) (it is an entry into a subroutine) (Raz column 6, lines 45-51).

29. Referring to claim 14, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising:

a. Determining a number of operands that are present on an operand stack at a time before the virtual machine instruction is executed (Linder column 11, lines 3-22) (It is inherent that this step will be taken in moving items from registers to memory and adjusting the SP for the instructions);

b. Determining a number of operands that are present on the operand stack at a time after the virtual machine instruction is executed (Linder column 3, lines 3-22); and

c. Inferring a number of shift operations required to maintain top-of-stack elements after the virtual machine instruction is executed (Linder column 3, lines 3-22).

30. Referring to claim 15, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 13, wherein the wherein the number of shift operations required after execution of the instruction is based on the number of operands present on the operand stack at a time before the instruction is executed and the number of operands present on the operand stack at a time after the instruction is executed (Linder column 11, lines 15-22).

31. Referring to claim 16, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 13, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table (Linder column 6, lines 39-47) (The translation is based on a static table. Through the table, it is known how many operands will be used and how many will be placed back on the stack, and based on that is how many items are transferred to memory.).

32. Referring to claim 17, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 12, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of an operand stack after execution of the virtual machine instruction (Linder column 11, lines 15-22).

33. Referring to claim 18, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 17, wherein the top-of-stack elements comprise a register stack (Linder column 11, lines 15-22).

34. Referring to claim19, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising execute a number of shift operations to replace top-of-stack elements to an operand stack (Linder column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

35. Referring to claim 20, Lindwer in view of Raz and in further view of Chan has the machine accessible medium according to claim 19, wherein the number of shift operations is based on a number of elements on the operand stack that are consumed by the virtual machine instruction and a number of elements that are produced by the virtual machine instruction (Linder column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

Response to Arguments

36. Applicant's arguments filed 31 August 2007 have been fully considered but they are not persuasive.

37. Applicant argue in essence on pages 8-9

> ...The change in phraseology from "shared execution code" to "cascading execution code" merely describes the already disclosed structure of the execution code and prevents overbroad interpretation of the term "shared."...

38. This has not been found persuasive. The term "shared" means "to participate in, use, enjoy, or experience jointly or in turns" (www.dictionary.com The American Heritage Dictionary of the English Language the 4th Edition © 2006). The term "cascaded" means "a succession of stages, processes, operations, or units (www.dictionary.com The American Heritage Dictionary of the English Language the 4th Edition © 2006)." As is seen by these definitions, the term "cascaded" has a significantly narrower definition than "shared", and, even with the cited paragraphs in the arguments, the Examiner cannot find any support in the original disclosure for narrowing the specification in such a significant manner by changing the term "shared" to "cascaded". Such a significant narrowing of the terminology needs to be reflected in the original disclosure, but it is not. The two terms are not equivalent and, even with Applicants' insistence, there was not previous indication that "shared" meant "cascaded". Changing the wording from "shared" to "cascaded" requires more evidence than Applicants' assertion that they mean the same, such as support from the original disclosure.

39. Applicant's argue in essence on pages 9-10

> ...Applicant reiterates that this is not the definition of threaded code as it used both in the present claims and in the pertinent art. In the present application, as well as well as in the art, threaded code does not refer to the concept of multi threading, in which a single processor runs multiple programs at the same time by

> switching between them rapidly, but to the compute programming concept of threaded code...

40. This has not been found persuasive. Threaded code and multithreading both refer to the concept of interleaving instructions from multiple programs to increase instruction throughput of the processor. The distinction that Applicants' arguments appear to be focusing on is the fact that "threaded code" is done by the programmer in software and "multithreading" is done by hardware. Whether it is in software or hardware is a design choice and not patentable subject matter, since they are logically equivalent. Please see Tanenbaum's Structured Computer Organization Second Edition ©1984. If there was any other distinction alluded to in the arguments, they are unclear to the Examiner. Also, any distinction must be made clear in the claims, since there is no associated clear, definite, and precise definition located by the Examiner in the claims.

41. Applicants' argue in essence on pages 10-11

> ...Chan refers to blocks "executing", only in terms of the order in which the instructions contained in the blocks will execute once the blocks have been compiled into an executable code.

42. This has not been found persuasive. Applicants' arguments appear to be arguing that the instruction contain machine instructions that are executed directly by the processor, and the instructions in Chan are intermediate program instruction that still needed to be decoded into machine instruction to be directly executed instructions. However, this is not in the claim. The claim merely states that the instructions are executed, which intermediate program instructions are executed. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., execution tiers or blocks means a plurality of machine level instructions that are directly executed by the

processor) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

43. Applicants' argue in essence on page 11

> ...Chan does not disclose wherein said cascading execution code comprises a plurality of tiers of execution code which are enterable at any tier...

44. This has not been found persuasive. The claim language being referred to states "...entering the cascading execution code at an entry tier indicated by the determined entry point and executing the entry tier and at least one tier below the entry tier." There is nothing in this claim language specifying that the tiers of execution are enterable at any tier. The claim language merely requires that the cascaded execution code is entered at an entry tier indicated by the entry point, which is taught in Chan as cited in the rejection above.

Conclusion

45. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

46. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

47. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:00am-4:30pm.

48. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

49. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Aimee J. Li

Aimee J Li
Examiner
Art Unit 2183

25 November 2007